Search
(courtesy of Google)

THE PracTEX Journal

The online journal of the TeX Users Group
ISSN 1556-6994

About *The PracTeX Journal*

General information
Submit an item
Download style files
Copyright
Contact us
About RSS feeds XML HTML

Archives of *The PracTeX Journal*

Back issues
Author index
Title index
BibTeX bibliography

Next issue
Approx. July 1st, 2008

Editorial board

Lance Carnes, editor
Kaveh Bazargan
Kaja Christiansen
Peter Flom
Hans Hagen
Robin Laakso
Tristan Miller
Tim Null
Arthur Ogawa
Steve Peter
Yuri Robbers
Will Robertson

Other key people

More key people wanted

Current Issue

2008, Number 2
[Published 2008-08-18]

Notices

From the Editor: In this issue; Next issue: LaTeX and TeX on the Web; Editorial: *A handy reference for Class & Style?*
Lance Carnes

Feedback
From the Readers

News from Around:
TUG2008; User group news; Math font videos
The Editors

Whole Issue PDF for PracTeX Journal 2008-2
The Editors

Articles

Class & Style — An introduction
The Editors

Go Game Positions with METAPOST
Wentao Zheng

Page Styles on steroids (or, memoir makes page styling easy)
Lars Madsen

Opinion: Enduring LaTeX documents
Lance Carnes

Columns

Travels in TeX Land: A bigger experiment with ConTeXt
David Walden

Ask Nelly:
How do I center only the last line of a paragraph?
How do I get the first and last entry of each index page in its header?
The Editors

Distractions: Fun packages — sudoku solvers
The Editors

Sponsors: **Be a sponsor!**

THE PracTEX Journal

From the Editor: In this issue; Next issue: LaTeX and TeX on the Web; Editorial: *A handy reference for Class & Style?*

Lance Carnes

- Comment on this paper
- Send submission idea to editor

In this issue
Next issue: LaTeX on the Web
Thanks
Editorial: A handy reference for Class & Style?

In this issue

This second issue of 2008 concentrates on a subject most LaTeX users love to hate: Class and Style packages. Hopefully, some of the articles here will ease this discomfort and encourage the use of packages.

The first article, *Class & Style — an introduction*, was written by the editors in response to a reader who asked for a *Compleat Idiot's Guide to Class and Style files* for new LaTeX users. We don't discuss files much, but show how to use the basic Class and Style features in LaTeX documents.

In *Go positions with METAPOST* Wentao Zheng writes about an ancient Chinese game now known best by its Japanese name, Go. The main technique he describes is the use of METAPOST to draw Go board positions. METAPOST is derived from the METAFONT program, which was originally used to design and create the Computer Modern fonts. Even if you don't play Go, the article shows the sort of pleasant graphic images that can be drawn with this tool.

Next, Lars Madsen describes some newly released features of a versatile document class called *memoir*. Following the theme of this issue, *memoir* is a Class, and was developed by LaTeX guru Peter Wilson. (Peter also devised a clever LaTeX sudoku solver. See this issue's Distractions column for more about this package.)

The last piece is an Opinion, "Enduring LaTeX Documents". Those of you who have used LaTeX for many years know the problems that occur when trying to format an old LaTeX document with a new LaTeX system — most often you get a different result, and sometimes new errors show up which weren't there before. The culprit? Class and Style packages! Can the LaTeX community solve this?

Dave Walden, in his *Travels in TeX Land* column, continues working on assembling a book of photography using the ConTeXt system. ConTeXt is a sister system to LaTeX but operates in a much different way. Where LaTeX has Classes and Styles that can be incorporated piece by piece as needed, ConTeXt is more of a monolithic system. It has a large set of tools and resources always at the ready, and the author uses those that work best for the current project.

In the Distractions column we offer a way to have fun with packages — for printing and solving sudokus. While you are using LaTeX to solve these addictive puzzles you may learn a few more things about Styles (packages).

We hope you enjoy this issue!

Next issue: LaTeX and TeX on the Web

We invite readers to submit articles on *LaTeX and TeX on the Web*. Do you publish typeset articles on the web or on a wiki? Do you create Web-based LaTeX and TeX training materials? Do you have a web site that lets users experiment with a running TeX system? Articles or notes on these or other projects with LaTeX or TeX on the Web are welcome. We encourage articles for this issue that are in html or otherwise formatted for the Web.

Submissions are due by September 30, 2008, and the issue will be published in October.

Send your article idea to the editors.

Thanks

Many people have collaborated directly or indirectly to the success of this electronic journal: the authors, the production editors, and the readers.

Thanks to my fellow issue editor, Yuri Robbers, and to all the production editors; and to others who proofread the articles and provided useful comments and feedback.

Editorial: A handy reference for Class & Style?

This showed up recently on the Slashdot site:

> *For many years I have been using LaTeX to compose scientific documents, but truly I am getting tired of its complexity. You have to install new packages for new features, ... and you need to be a LaTeX Jedi master to create a new document class. I'm looking for a document processor ... that is a viable replacement for LaTeX, possessing all of its advantages — consistency between text and math text, automated cross references, direct PDF creation, etc. ...* [Full quotation]

This person voices frustrations many of us have as LaTeX users. However, the printed results from LaTeX are so fine that we put up with the complexity. Only a handful of LaTeX users create new document classes so the "LaTeX Jedi master" name is fitting, but packages are used by nearly everyone. I wish this person luck finding a replacement document processor — LaTeX (TeX) is simply the best tool for document composition.

But the person has a good point about the complexity of using packages to incorporate new features. When doing my main work I'm willing to spend some time learning new techniques, but when I use LaTeX it's a different matter. Writing documents with LaTeX isn't my main occupation, and I don't use it often enough to have a lot of commands at my fingertips. When I need an answer, it usually involves a lengthy hunt for information. I consult the few LaTeX books on my shelf, then look online, then experiment for a while. Following is an illustration of how many LaTeX users go about creating a document.

If you are a regular LaTeX user (that is, you are not a LaTeX expert), your document source probably looks something like this:

```
% My usual document source
\documentclass{article}

\usepackage{times}

\begin{document}
\title{A paper}
\author{A. U. Thor}

...

\end{document}
```

Like most LaTeX authors I have a collection of previous documents that begin with something like the fragment shown above. When it's time to write a new document I make a copy of an old one and then change the content. (If this isn't similar to the way you operate, you're probably a LaTeX expert and can skip reading the rest of this.) Life goes on, and I get my documents written.

Then one day I need a new feature. I saw a LaTeX article somewhere (perhaps in *The PracTex Journal*?) that had clickable links that take you to another part of the article or even to a web page. This seemed like a really nice feature, and wouldn't it be nice if I could use this in my documents?

If I'm in an adventuresome mood and have a few hours to spare I'll try to figure out this new feature. Some of the articles in *The PracTex Journal* have the source document available and I can borrow the authors' tricks. It will probably take some time to figure out, but once mastered, it's a new arrow in my quiver. I add the new trick to my usual document source:

```
% My usual document source
\documentclass{article}

\usepackage{times}
\usepackage{hyperref} % make links in my document

\begin{document}
\title{A paper}
\author{A. U. Thor}

...
\href{http://www.mypage.net}{My Web site} % a link!
...

\end{document}
```

The next time I need to write a new document I can copy this one and start anew. The new tricks I picked up are included and ready to be used again.

The downside is that each time you add a new technique to your "usual document source", it may take an hour or two or more. If you like experimenting with LaTeX this is no problem. But for those who want to spend their time writing rather than fiddling around with LaTeX, the extra hour or two needed to figure out each new feature is prohibitive, and you just do without.

A year or so ago *The PracTex Journal* received an article from an engineering Ph.D. student, Lapo Mori. While writing he must have found he was spending a lot of time researching LaTeX features. Shortly after submitting his article, he proposed that there should be a *LaTeX-pedia*. That is, an online resource where an author can easily find useful information about LaTeX, and examples of LaTeX documents. Lapo proposed this project to his friends at the Italian Users Group (GuIT), and it has become one of the group's long-term goals. I haven't spoken to Lapo recently and don't know the state of this project, but it certainly seems like one that needs to happen sooner than later.

There are currently several good online LaTeX resources. In my browser's bookmarks I've collected about 100 such sites. Each has its advantages and maybe a few shortcomings. Still, when I want to know about a package or feature, I most often use a search engine and type something like, `LaTeX format url`, to find out how to typeset a url. Using this method I often land at one of the sites I've already bookmarked, and sometimes I find a new one, but it's usually hit or miss.

Rather than constantly looking through a variety of sites it would be nice to have a single site where I could start my search. Even if this site were just a well-maintained index to other currently available sites it would be a great resource. And who knows, over time it could even become a self-contained complete reference.

The LaTeX-pedia or someting similar would be a valuable resource. I'd certainly use it, and it might even satisfy the LaTeX-weary Slashdot person.

Lance Carnes
Editor

THE PracTEX Journal

Feedback

From the Readers

- Comment on this paper
- Send submission idea to editor

TPJ

I am a rather new user to Latex and installed in on my computer some month ago. I have to admit that the learning curve has been steep. The Mori paper in issue 2008-1, http://tug.org/pracjourn/2008-1/mori/ was indeed very useful and informativ. Beginners as me do not necessary need all details and proof why, but good recommedations. You experts know what is working and why, and how to modify everything. For the time being, I need information what is working. Therefore, this article was exellent.

Kind regards,

Sveinung Heggen, Norway

[Thanks for your feedback! We're glad the paper was useful to you. — The Editors.]

TPJ

Sir,

I don't know much about pstricks but I used Asymptote to draw the same curve (or so it seems to me) and I got the following picture. It looks a bit different from what I see in the distractions in issue 2008-1, http://tug.org/pracjourn/2008-1/distract/. Do you see any reason? Am I drawing a different curve?

My asymptote code is:

```
import graph;

currentpicture=new picture;

size(10cm,10cm);
real R, r, d;

R=100;
r=2;
d=80;

real x(real t){return (R-r)*Cos(t)+d*Cos(((R-r)/2)*t);}
real y(real t){return (R-r)*Sin(t)-d*Sin(((R-r)/2)*t);}

draw(graph(x,y,0,360,n=10000,operator ..));
```

```
shipout("ipertrocoide");
```

Maurizio Vianello, Italy

[Dear Maurizio, it seems you are using 10000 evaluation points whereas I used only 1000. This might explain the small inaccuracies in my version which lead to every line being drawn multiple times slightly displaced. Your version seems the better one. — Yuri Robbers]

TPJ

Just read (skimmed, to be honest) Ista Zahn's article "Learning to Sweave in APA Style" in issue 2008-1, http://tug.org/pracjourn/2008-1/zahn/. It reminds me of something we're trying to do in computational linguistics, where we're using Literate Programming to embed linguistic rules in a document. We're actually using DocBook, but we use dblatex to convert the DocBook document into LaTeX form, and XeTeX to format that into a printable or displayable document.

One of the problems we've encountered is that of long tables. It seems to me that LaTeX is very good about formatting tables--if you know in advance how much room the table is going to occupy on the page. If you don't know that, then you can wind up with a table that runs off the end of the page, or tables that are split when they should have been floated, depending on which table package you use. There doesn't appear to be a single table package that handles both floating tables, and tables that are longer than a single page.

In our case, we can look at the final result and change the kind of DocBook table we use, and as a result the kind of LaTeX table that dblatex gives us. But that seems like it shouldn't be necessary; there ought to be a table package that looks at the size of a formatted table, and decides whether to leave it where it is, float it, or break it over a page (or perhaps both float it and break it over pages). This is surely not too hard for a computer to do (although maybe it was too hard for most people's computers to do fifteen or so years ago, when the table packages were written).

Even when we choose the right kind of table for a particular document, later changes to the document (or changes to the page size) may reveal other places where we should have chosen a different kind of table.

In the case of tables that are automatically produced by sweave, this seems like an even more severe problem. The table is not produced by hand--that's the whole point of using sweave--so there is no direct control over the length of the table.

So my question: On the assumption that the tables produced by sweave are normally floated, does it ever happen that sweave produces a table that is too long to fit on a single page? And if so, does sweave automatically compensate by using a different table package (one that allows multi-page tables) for that table?

If not (and I'm guessing it doesn't), isn't it time to have a table package that can float a table, or split a table over a page boundary, and figure out on its own when to do one or the other?

Mike Maxwell, USA

[I don't feel that I have much insight to offer on this issue. However, I'm happy to share what very little I do know.

I agree that it would be very nice to have a table environment that does "the right thing" depending on the size of the table. It is true that the xtable package in R defaults to a regular floating table environment, and that this will sometimes produce tables that do not fit on the page. The xtable R package can be used with the longtable packge (it can also be used with ctable, and sideways) but I am not aware of a way to have the program intelligently decide which environment to use. In my own personal use this is generally not too troublesome, because I rarely make use of long tables, and when I do I usually know ahead of time that the table will be too long to fit on a single page. So while I agree that it would be useful to have a package smart enough to place tables appropriately no matter what size they are, unfortunately I haven't the slightest idea how to go about doing this. Please let me know if you figure out a way!

Sincerely,

Ista Zahn]

TPJ

THE PracTEX Journal

News from Around: TUG2008; User group news; Math font videos

The Editors

- Comment on this paper
- Send submission idea to editor

International TUG Meeting

TUG 2008 was held in Cork, Ireland July 20-24. The conference offered technical presentations, LaTeX workshops, and, of course, a whiskey tasting. Multimedia presentations from the conference are available courtesy of River Valley Technologies.

User group news

Editor Apostolos Syropoulos announced that the Greek TeX Friends released the latest online issue of their journal, Eutypon.

The latest issue of The Asian Journal of TeX (AJT), vol.2, no.1, is available online. This is published by the Korean TeX Society headed by Jin-Hwan Cho.

The Italian TeX Users (GuIT) released their print journal, *Ars TeXnica*, Number 5. To order a copy, which comes with a plastic pica ruler, visit their site. GuIT will be holding their **annual meeting** in Pisa on October 18, 2008 in Pisa.

The German TeX Users (DANTE) just released the latest issue (60pp) of *Die TeXnische Komödie*. Their fall conference will be held on September 13, 2008 at Tüaut;bingen University.

Math font videos

PCTeX recently produced three videos by Michael Spivak on the development and features of MathTime fonts. He has spent over 20 years developing these popular fonts, and describes the improvements over traditional math fonts.

THE PracTEX Journal

Class & Style — An introduction

The Editors

- Comment on this paper
- Send submission idea to editor

Class & Style

An introduction to LaTeX document classes and styles

by The Editors

A reader wrote, *An article that I would certainly read with enormous interest would have the title "Compleat Idiot's guide to using .class and .sty files". In other words a blow-by-blow description of how to use those types of files for someone very new to LaTeX.*

Let's start with *class* and *style* in LaTeX. Consider this short document:

```
\documentclass{article}

     \begin{document}

     A Pascal (Pa) is equal to a Newton per square meter, $N/(m^2)$.

     \end{document}
```

Class. This document uses the *article* document layout, also called the article *documentclass*. A documentclass defines the general appearance of a document: it sets margins, the type size for section heads, the placement of page numbers, and other document design features.

Style. A LaTeX *style* introduces new functionality and commands. As an illustration, let's add some style changes to make the document above more professional looking. The scientific units, `$N/(m^2)$`, as they stand will format, but not correctly for a scientific journal. There are standards for this, and you could look them up and try to get LaTeX to follow the rules. Fortunately, there is an easier way. A LaTeX user developed a package or *style* called `SIunits` that will correctly format scientific units.

If you use a *style* package your LaTeX document will look like the following:

```
\documentclass{article}

     \usepackage{SIunits}  % Scientific units package

     \begin{document}

             A Pascal (\pascal) is equal to a Newton per square meter,
             \newton\per\square\metre.

     \end{document}
```

Notice that a new command, `\usepackage{SIunits}`, appears in the document preamble. Also notice that now you can use some new commands: `\pascal`, `\newton`, `\per`, etc. These were introduced by the style package `SIunits`. If this document is

formatted, you will see that the scientific units are formatted correctly. (It's nice that someone else figured this out so that you don't have to!)

In the above example, we have shown how to format a LaTeX document using a `documentclass`, and how to add new LaTeX commands by the use of a *style* package.

Class & Style. To tie this in to the theme of this issue, *Class & Style*: the `article` documentclass is found in a file called `article.cls` and is sometimes called a *Class* (`.cls`) file; the package `SIunits` is found in a file called `SIunits.sty` and is sometimes called a *Style* (`.sty`) file.

Changing the Class

One way to change the appearance of a LaTeX document is through the `\documentclass` command options. To change the text font size, for example, add an option to `\documentclass`:

```
\documentclass[12pt]{article}
```

This will change the text size from 10 point (the default) to 12 point.

Another way to change the appearance of a document is to change to another documentclass. For example, change `\documentclass{article}` to

```
\documentclass{memoir}
```

Format the document and notice that the overall appearance is slightly different. Some alternate Classes are shown in the TeX Catalogue. Note that the TeX Catalogue has an easy way to view documentation for Classes and Styles.

Finding Styles

In the above example we used a package `SIunits` to format scientific units. You can find this in the TeX Catalogue Topic Index under *Science* and then *Typesetting Physical Units*.

There are more than 2,000 LaTeX Style packages. The TeX Catalogue is a good starting point to locate the package you need. Another resource is CTAN. You can also try an online search engine. Try typing `LaTeX physical units package` into Google.com, for example, and it will find a few references to the `SIunits` package.

More on Class & Style

Jim Hefferon, What I Wish I Had...When I Was A Lad: Using LaTeX Resources
Jim Hefferon, CTAN for Starters
Jim Hefferon, Minutes in Less Than Hours: Using LaTeX Resources

Retrieved from "http://wiki.pctex.com/index.php/ClassStyle"

Page generated August 24, 2008 ; TUG home page; search; contact webmaster.

THE PracTEX Journal

Go Game Positions with METAPOST

Wentao Zheng

Abstract

This article introduces a method of drawing Go game positions with MetaPost. It begins with how the Go game is modeled in the MetaPost language, then explains the detailed implementation, and ends with some examples of Go game positions.

Wentao Zheng is a researcher at the IBM China Research Laboratory. His research interests are focused on interactive graphics and visualization. You can reach him at zhengwt@cn.ibm.com

Date: 06/17/2008

IBM assents to the assignment of any copyright ownership that IBM may have in the employee-written material in the compilation of the work:

Go game positions with Metapost

by: Wen Tao Zheng

to the publisher of: PracTeX Journal

This assent is contingent on the publication of the work by the publisher identified above. IBM and the author shall each have the royalty-free right in any work incorporating the material on which IBM assented to an assignment of copyright, to make and distribute (1) abstracts of the work to a selected listing of the scientific community prior to publication by the publisher, (2) copies of the work and its derivatives for IBM internal purposes prior to or after publication by the publisher, and (3) individual copies to members of the scientific community outside of IBM in response to specific requests or for peer communication.

- PDF version of paper
- Article sources
- Comment on this paper
- Send submission idea to editor

The PracTEX Journal, 2008, No. 2
Article revision 2008/7/24

Go Game Positions with METAPOST

Wentao Zheng

Email zhengwt@cn.ibm.com
Address IBM China Research Laboratory

Abstract This article introduces a method of drawing Go game positions with METAPOST. It begins with how the Go game is modeled in the METAPOST language, then explains the detailed implementation, and ends with some examples of Go game positions.

1 Introduction

Go is a board game that originated in ancient China in the 4th century BC. In Chinese, it is known as *weiqi*, which means "board game of surrounding". It is commonly known in the West by its Japanese name *igo* (the Japanese reading of its Chinese name), because early western players learned the game from Japanese sources.

Go is considered the most complex game of all, and has a long history of academic study. Often the Go game graphics in research papers are drawn with low quality. This article introduces a method of drawing Go game positions with METAPOST that can produce flexible and high quality Go game graphics.

This article is organized as follows: section 2 introduces modelling the Go game in METAPOST; section 3 describes the detailed implementation of the method; and section 4 shows some examples of using this method to draw Go game graphics.

2 Modeling Go in METAPOST

Go is played by two players alternately placing black and white stones on the vacant intersections of a 19×19 grid board (see Figure 1). The object of the game is to control a larger part of the board than the opponent. In a game position, if

there are no adjacent intersections for a stone or a group of stones, it is considered as captured and the stones are removed from the board. For more information about Go game rules, please see 'Go (board game)' from Wikipedia [2].

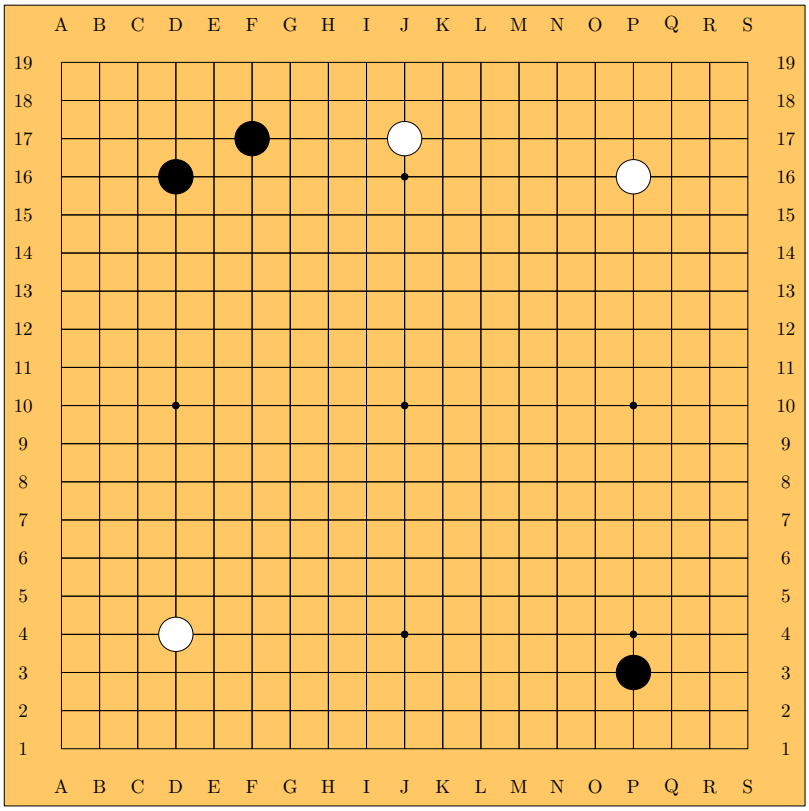


Figure 1: A standard Go game board with stones

In the initial Go game position, there are no stones. To begin a game, the first player puts a black stone, and then the opponent puts a white stone. These two steps are repeated until no more stones can be placed. On any turn a player may choose to skip his or her play. From a programming perspective, this process can be described as follows:

1. Initialize the Go board
2. Black player puts a stone (optional)
3. White player puts a stone (optional)

4. Repeat steps 2 and 3 until no stones can be put on board

In each stone-putting action, we check whether the stone will capture the opponent's stones or be captured (suicide). We model a Go game position in METAPOST as four major routines (components)

- Board initialization
- Stone put
- Capture check
- Board display

We will describe each component in detail in the next section.

3 Implementation

3.1 Board initialization

To provide a flexible way to initialize the board, we use several internal parameters (Figure 2). These parameters control how a Go board will be rendered.

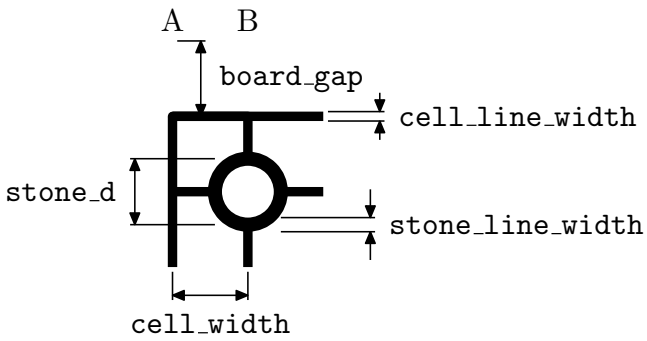


Figure 2: Go board parameters.

board_color the background color of the Go board

cell_width the width of the board grid

cell_line_width the width of line used in drawing the board

stone_d the diameter of the stone (default as 0.9 × `cell_width`)

stone_line_width the width of line used in drawing the stone outline

board_gap the gap between board edges and board labels

board_size the size of the board (default as 19)

The board status is recorded in METAPOST as a two-dimensional array

```
numeric mem[][]
```

Each element in the array represents a Go board intersection, and the value can be

-1 white stone

1 black stone

0 empty

There is a routine called

```
init_board
```

that is used to initialize a Go board with all intersections being empty.

3.2 The Stone-put

Stones are put on the board by specifying their coordinates. Each coordinate has two values, horizontal and vertical, as shown in Figure 1. The horizontal coordinate is a character starting from the letter 'A', and the vertical coordinate is an integer number starting from 1. If the value is less than 1 or 'A' or greater than `board_size`, it is considered as invalid.

There are two routines for putting stones

put_black put a black stone at specified coordinate

put_white put a white stone at specified coordinate

The coordinates used to identify the positions of stones are useful for human players. But METAPOST does not know where to put the stones directly based on player-oriented coordinates, such as 'A5'. The secret behind this action is a transformation that converts them into METAPOST-understandable coordinates, such as `(20pt,80pt)`. The latter coordinate form is used in the board graphics generation routine.

3.3 Capture check

After putting a stone on the board, it is necessary to check whether the stone captures the opponent's stones or can be captured (suicide). Generally, there are three rules:

- If the stone-put is not a suicide and captures some of the opponent's stones, then the captured stones are removed from board. (Figure 3(a))
- If the stone-put is a suicide and captures none of the opponent's stones, then the stone is removed from board. (Figure 3(b))
- If the stone-put is a suicide and captures some of the opponent's stones, then the stone is reserved and the captured stones are removed from board. (Figure 3(c))

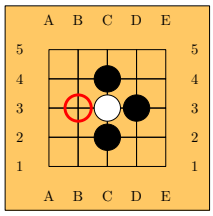
After each move, there are two steps to check for captured stones. In the first step, the routine

```
_capture <coord>
```

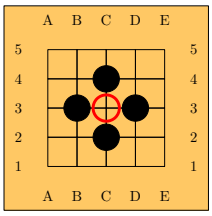
is used to check whether the stone at the specified coordinate will capture the opponent's stones. If so, the captured stones are removed immediately. In the second step, the routine

```
_suicide <coord>
```

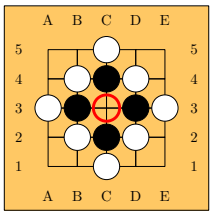
is used to check whether the stone at a specified coordinate is a suicide. If so, the stone is removed. By checking a move in this order, we can ensure that a stone which is both a suicide and captures other stones is reserved and that those captured stones are removed.



(a) Capture (put a black stone)



(b) Suicide (put a white stone)



(c) Capture and suicide (put a white stone)

Figure 3: Three rules for capture check (red circles to put stones)

3.4 Board display

To graphically display a Go game (including board and stones over a sequence of moves), there are two different approaches. The first one is **programmatic**. Go game graphics are generated by calling the routine

```
display_board
```

in METAPOST code. The second approach is **script-driven**. You write a script file which contains information about where to place the stones step by step. By calling the routine

```
script <file_name>
```

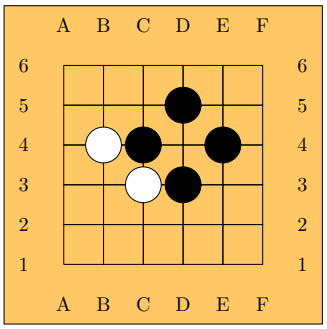
Go game graphics will be generated automatically for each step. In the script file, stone-put operations are separated into lines. The action is formatted as three parts: color, horizontal coordinate, and vertical coordinate. For example, "B A 2" means put a black stone at position (A,2).

4 Examples

In this section, two examples of using our method to generate high quality Go game graphics are presented.

The first one uses the programmatic approach.

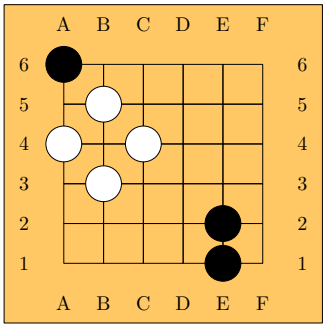
```
beginfig(1);
  set_board_size 6;
  init_board;
  put_black(C,4);
  put_white(D,4);
  put_black(D,3);
  put_white(C,3);
  put_black(E,4);
  put_white(B,4);
  put_black(D,5);
  display_board;
endfig;
```



This second example uses the script-driven approach.

```
%%% content in script file "test.go"
B A 6
W B 5
B B 4
W A 4
B E 2
W B 3
B E 1
W C 4

%%% content in MP file "test.mp
set_board_size 6; script "test.go";
```



5 Summary

This article introduced Go game positions with METAPOST. Motivation, general design and implementation of the method were described in separate sections. Additionally, some examples of how to use the method to generate high quality Go game graphics were shown. There are still some points to improve, such as supporting *ko* rule [2].

Acknowledgments

I'd like to thank all those who helped me on writing this article. They are Lance Carnes, Yuri Robbers and two anonymous reviewers.

References

[1] John Hobby, "METAPOST: A User's Manual".

[2] Go (board game), http://en.wikipedia.org/wiki/Go_(board_game)

THE PracTEX Journal

Page Styles on steroids (or, memoir makes page styling easy)

Lars Madsen

Abstract

Designing a page style has long been a pain for novice users. Some parts are easy, others need good LaTeX knowledge. In this article we will present the Memoir way of dealing with page styles including new code added to the recent version of Memoir, that will reduce the pain to a mild annoyance.

We will end the article with a series of common scenarios and how to solve these.

Lars Madsen holds a master's degree in math from the Department of Mathematical Sciences, University of Aarhus, Denmark, where he is also currently employed as a programmer doing general user support, including quite a lot of LaTeX support.

Lars is a regular on comp.text.tex and the Danish TeX User Group's mailing list. He is currently the vice president of the Danish TeX User Group DK-TUG. As he witnesses a lot of 'normal' LaTeX users, Lars is a bit concerned on how new users learn LaTeX. As a result of this, he is currently working on the third edition of his 300+ pages Danish LaTeX introduction. He can be contacted at `daleif@imf.au.dk`

- PDF version of paper
- Article sources
- Comment on this paper
- Send submission idea to editor

The PracTEX Journal, 2008, No. 2
Article revision 2008/7/24

Page styles on steroids or *memoir* makes page styling easy

Lars Madsen

Email daleif@imf.au.dk
Website http://home.imf.au.dk/daleif/
Address Department of Mathematical Sciences
Faculty of Science, University of Aarhus
Denmark

Abstract Designing a page style has long been a pain for novice users. Some parts are easy; others need strong LaTeX knowledge. In this article we will present the *memoir* way of dealing with page styles, including new code added to the recent version of *memoir* that will reduce the pain to a mild annoyance.

We will end the article with a series of common scenarios and how to solve them.

Prerequisites

The prerequisite for this article is access to a recent version of *memoir* (that is, *memoir* v1.61803, released in June 2008 or newer; we will refer to v1.61803 as *memoir* 2008).

Preliminary information

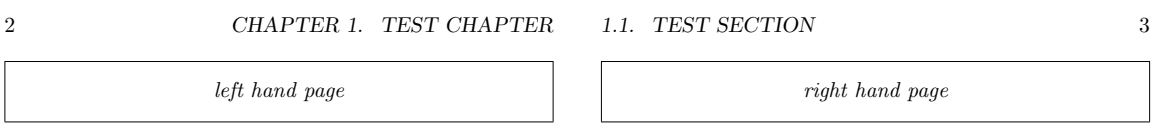
Examples in this article that show a graphical result have been created using the *memoir* class, displaying a two-sided setup. The examples will show a left-hand (even) page and a right-hand (odd) page, with a dummy text block leaving all attention on the headers and the footers. We have also added a dummy chapter, section, subsection and subsubsection, in order to show their effect on the page styles (header information). One can think of the material as page two and three of a chapter that starts out as

Copyright © 2008 Lars Madsen
Permission is granted to distribute verbatim or modified
copies of this document provided this notice remains intact

```
\chapter{Test chapter}
\section{Test section}
\subsection{Test subsection}
\subsubsection{Test subsubsection}
```

Source code in examples will be shown in a green color; code in the text and showing syntax will be shown in a red color; and page styles will be shown in cyan. Here, for instance, is an example displaying the default page style in *memoir*, known as headings:

```
\pagestyle{headings}
```



In this article we will handle only the *two-sided* setup. The *one-sided* counterpart comes rather easily; just remember that whenever the `oneside` option is in effect, the page style design for all pages will be the one corresponding to an odd page.

1 The basics

In this section we will summarise the basics concerning page style handling, plus we will list some of the challenges that we face compared to the standard classes (`book`, `report` and `article`).

1.1 What is a page style?

In LaTeX the *page style* refers to the part of the page building mechanism that attaches the headers and footers to the actual page. Technically, in a two-page setup, the headers and footers are inserted via four macros `\@evenhead`, `\@oddhead`, `\@evenfoot` and `\@oddfoot`. Activating a particular page style will redefine these four macros to do whatever the style is designed to.

> Remark. *Please note that the actual vertical placement of the headers and footers is not a matter of the page style. This depends on the dimensioning of the page layout.*

1.2 Activating a page style

A page style can be activated in two ways: globally and locally (here locally means *on this page only*).

```
\pagestyle{⟨style⟩}
\thispagestyle{⟨style⟩}
```

`\pagestyle{⟨style⟩}` activates ⟨*style*⟩ from the *next page and onwards*, while `\thispagestyle{⟨style⟩}` enables ⟨*style*⟩ on this page only.

The page style used on a particular page is decided as follows. If any `\thispagestyle` macros have been issued on this page, we use the last one. If no `\thispagestyle` macros have appeared, we use the global page style (i.e. the style recorded to have been set via a `\pagestyle` at the *previous* page break/start).

1.3 Making material available for the headers and footers

In the header and footer we can add static information, i.e. define the four macros to just display some non-changing string. But we also have access to more dynamic information such as the page number (via `\thepage`), and via special macros we can provide access to all sectional titles.

To make titles available we make use of so called *marks*, `\leftmark` and `\rightmark`, which are available on all pages. These can be set, in macros,[1] using

```
\markboth{⟨for left mark⟩}{⟨for right mark⟩}
\markright{⟨for right mark⟩}
```

There is no separate `\markleft`. For the page building mechanism the content of the `\leftmark` and `\rightmark` is determined as follows: `\leftmark` contains the *last* ⟨*for left mark*⟩ issued before the end of the page, while `\rightmark` contains the *first* ⟨*for right mark*⟩ issued on the page. If none have been issued on this page the most recently defined one from former pages will be used.[2]

1. The user may also use them manually in the text, though this might annoy copy editors.
2. Note that in some circumstances some confusion may arise concerning the content of the marks, see [4], section 4.3.4, page 218–219, although the marks work without any problems in most normal chapter-based documents.

To ease things, all sectional macros have been equipped with a special macro, named \⟨*sectional macro*⟩mark, e.g. \sectionmark, which will be given a title as an argument. These sectional marking macros initially do nothing, but they can be redefined to use \markboth or \markright.

REMARK. *Please recall that the sectional macros in LaTeX support up to two titles, i.e.*

\chapter[⟨*for head and toc*⟩]{⟨*for text*⟩}

where ⟨*for text*⟩ *will be used for all if* ⟨*for head and toc*⟩ *is missing.*

In memoir we extend this even further by supporting the syntax above and supplementing it with

\chapter[⟨*for toc*⟩][⟨*for head*⟩]{⟨*for text*⟩}

In addition to the sectional macros, several other macros add material to the marks, including the table of contents (adding \contentsname), the index (adding \indexname), the bibliography, etc.

1.4 Challenges

As most LaTeX users know, the standard classes (book, article, report), are not particularly flexible. There are basically no configuration interfaces at all; one needs to resort to external packages.

In the area of page styling, the situation in the standard classes is no different. The standard classes have several annoying features regarding the material used in the page styles.

(1) All mark content provided by \chapter, \section, etc. are hardwired to be in upper case.
(2) Marks set by macros and environments such as \tableofcontents, the bibliography and the index are all in upper case. This feature cannot easily be changed without having to copy a large number of code lines and then only changing one or two lines of that code.
(3) In case your setup allows a blank page before starting a new chapter (default in the book class), that »blank« page is not really blank; it has the same page style as all other regular pages, which looks odd if there is no other content

on the page. The solution is to manually change \cleardoublepage in order to automatically make these pages truly blank.

(4) There is no easy way to redesign the look of the footers and headers.

(5) Creating truly flexible sectional marking macros — \chaptermark, \sectionmark, etc. used internally by \chapter etc. — demands a lot of LaTeX knowledge from the (novice) user.

The fancyhdr package can handle several of these problems but with some costs. As far as I know it does not provide an easy solution to item (5).

2 Page styling in memoir

In *memoir* we deal with two types of page styles, *aliases* and *regular* page styles. An alias page style is a style that just refers to another style, i.e. if the style alias is an alias for the empty style, then \pagestyle{alias} is the same as \pagestyle{empty}. The advantage of having aliases is that we can, for example, provide the first page of a chapter with its very own page style without the need to create a specific style for it. We just let the chapter style point to some other style, say, plain; and if we later decide that we want a different look for the chapter title page, then we just let chapter point to some other page style.

2.1 How page styles work in memoir

As mentioned earlier the headers and footers are added to the page via the four macros \@evenhead, \@oddhead, \@evenfoot and \@oddfoot. In *memoir* a given page style ⟨*style*⟩ consists of a number of specially created macros, all associated with the name ⟨*style*⟩, and activating ⟨*style*⟩ causes LaTeX to rebuild those four macros using the current content of the macros associated with ⟨*style*⟩.

The *memoir* page style interface interacts with the special macros associated with ⟨*style*⟩.

> Remark. *There is one important thing to remember. Whenever one has changed a page style in memoir (we will cover this shortly), that page style has to be activated in order for the changes to take effect. For a main document style, this means that you have to issue* \pagestyle{⟨*style*⟩} after *you have made any changes to the style. The reason is that we use the contents of the* ⟨*style*⟩ *macros, not the macros themselves.*

For styles that are just used on local pages, like the chapter style, no reactivation is necessary.

2.2 Creating a new pagestyle

In *memoir* we use one of the following macros to create a new page style:

```
\makepagestyle{⟨name⟩}
\copypagestyle{⟨new name⟩}{⟨old name⟩}
\aliaspagestyle{⟨alias⟩}{⟨original⟩}
```

`\makepagestyle{⟨name⟩}` will create a new page style named ⟨*name*⟩ with all of its corresponding internal macros initialised to do nothing.

`\copypagestyle{⟨new name⟩}{⟨old name⟩}`, does the same, but initialises the internal macros to use the contents of the internal macros which currently form the style ⟨*old name*⟩. This is very handy, as it allows you to copy a specific style, and change the headers in the copy while maintaining the look of the original footers.

`\aliaspagestyle{⟨alias⟩}{⟨original⟩}` does *not* create internal macros corresponding to its own name. As a consequence, changing an alias page style using the macros we will present in the next section will have unexpected results. The alias uses nothing from ⟨*new name*⟩, so the changes never go into effect. Basically, do *not* change an alias page style directly; if you need to change an alias style, then first overwrite it (using `\copypagestyle`) with a style you would like to use as a base.

REMARK. *Note that all three macros will happily* OVERWRITE *an existing page style without warning.*[3]

* * *

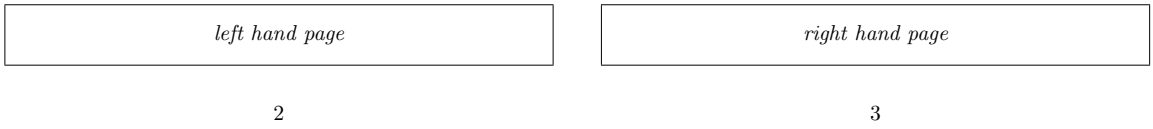
Here is a list of the regular page styles that are provided by *memoir*:

```
\pagestyle{empty}
```

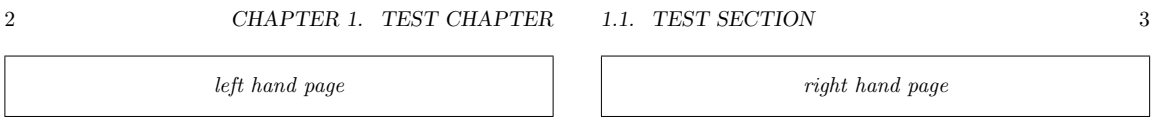
left hand page	*right hand page*

3. A handy feature as we will see later.

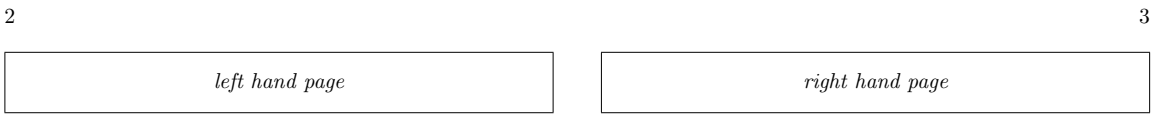
```
\pagestyle{plain}
```




```
\pagestyle{headings}
```

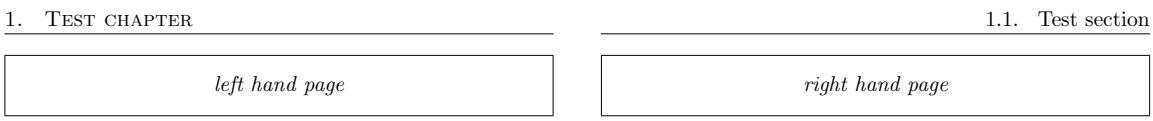



```
\pagestyle{myheadings}
```

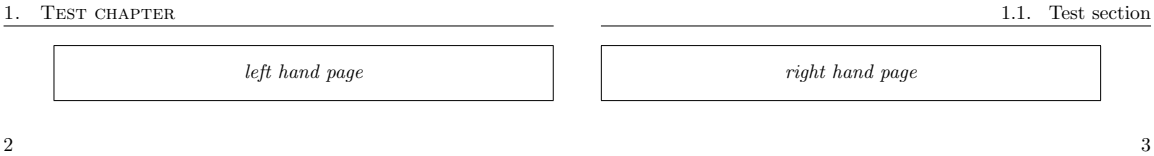


myheadings does not automatically write anything to the headers besides the page number. Users can add the text themselves. This style is handy for writing conference proceedings, where one often sets the headers to show the author names and the title of the contributed article.

```
\pagestyle{ruled}
```

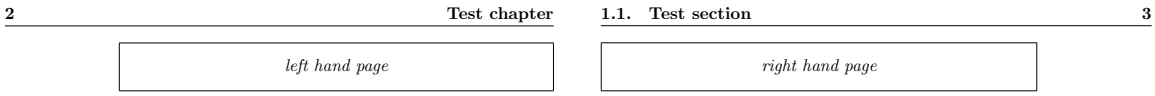



```
\pagestyle{Ruled}
```



Note how this style protrudes into the outer margin.

```
\setlength{\headwidth}{\textwidth}
\addtolength{\headwidth}{\marginparsep}
\addtolength{\headwidth}{\marginparwidth}
\pagestyle{companion}
```



The companion style mimics the style used in *The LaTeX Companion* by Mittlebach et al, [4]. The design of [4] uses very wide outer margins for notes and such.

* * *

The provided aliases page style in *memoir* are:

chapter	→	plain	title	→	plain
part	→	plain	titlingpage	→	empty
afterpart	→	empty	book	→	empty
cleared	→	empty	afterbook	→	empty
indextitlepagestyle	→	chapter			

The after... styles are the back side of the part and book title pages, see [1]. The indextitlepagestyle has been added in order to make it easy to create very special effects in the index; see for example the index of [3]. title is the page style issued by `\maketitle` whereas titlingpage is the page style for the *first* and *last* page of the `titlingpage` environment.

2.3 Setting up the design of a page style

Designing a page style is a two-step process. First there is the visual design, and second there the preparation of macros that provide the information that goes in the headers and footers. In this section we will look at the design part.

The main macros for dealing with the design of the headers and footers are:

```
\makeevenhead{⟨style⟩}{⟨left⟩}{⟨center⟩}{⟨right⟩}
\makeoddhead{⟨style⟩}{⟨left⟩}{⟨center⟩}{⟨right⟩}
\makeevenfoot{⟨style⟩}{⟨left⟩}{⟨center⟩}{⟨right⟩}
\makeoddfoot{⟨style⟩}{⟨left⟩}{⟨center⟩}{⟨right⟩}
```

\makeheadrule{⟨*style*⟩}{⟨*width*⟩}{⟨*thickness*⟩}
\makefootrule{⟨*style*⟩}{⟨*width*⟩}{⟨*thickness*⟩}{⟨*skip*⟩}
\normalrulethickness

The first four macros are easy to understand. They refer to the headers and footers, where one can place material to the left, in the center, or to the right. For example, to create a header which displays titles from the (sub-,subsub-)section in the header for right hand (odd) pages, one could use

```
\makeoddhead{mystyle}{\itshape\rightmark}{}{\thepage}
```

provided that \⟨*(sub-,subsub-)*⟩sectionmark provides information for \rightmark.

One simply specifies the style one wants to change, and then specifies the contents for left, center, and right for that particular part. Traditionally, for a design that supports chapters, one has the chapter title sent off to the \leftmark, whereas all other titles would be sent off to \rightmark. In this manner, a widely used setup is:

```
\makeoddhead{mystyle}{\itshape\rightmark}{}{\thepage}
\makeevenhead{mystyle}{\thepage}{}{\itshape\rightmark}
```

The standard headings is like this; it just uses \slshape instead of \itshape.

To add lines we use either \makeheadrule or \makefootrule. In most cases the ⟨*width*⟩ will be equal to \textwidth and for the ⟨*thickness*⟩ one can use the value set by \normalrulethickness (usually 0.4pt), which matches most other rules set by standard LaTeX. The ⟨*skip*⟩ part will usually be just \footruleskip, see [1].

As can be seen with the Ruled and companion page styles, one can also create styles that protrude into the outer margins; these can be set using \makerunningwidth{⟨*style*⟩}{⟨*width*⟩} and \makeheadposition{⟨*style*⟩}{⟨*eheadpos*⟩}{⟨*oheadpos*⟩}{⟨*efootpos*⟩}{⟨*ofootpos*⟩} as explained in [1]. Users who would like to use these features are encouraged to have a look in [1] and inside the *memoir* source code (`memoir.cls`) to see how these macros work.

2.4 Setting up content providers

As mentioned earlier all sectional macros (\chapter, \section, etc.) have the ability to make their title available for the page styles via \chaptermark{⟨*text*⟩},

\sectionmark{⟨*text*⟩}, etc. which are built into the sectional macros. In addition, *memoir* also adds some simple mark macros that are associated with other constructions that may have their own special headers. These include the table of contents (\tocmark), list of figures/tables (\lofmark), the index (\indexmark) and the bibliography (\bibmark).[4]

The *memoir* class includes an extra function one can use to redefine what the sectional and plain marks should do.

\makepsmarks{⟨*style*⟩}{⟨*code*⟩}

The ⟨*code*⟩ will be executed when ever the ⟨*style*⟩ is activated.

* * *

It has always been hard to create generally applicable sectional marks. One needs to take into account:

1. If sectional numbering for this level is switched off, then no sectional number should be printed in the content provided for \leftmark and \rightmark.
2. Outside the main matter numbers should never be added.
3. One might want to add some sort of formatting to the number when it is printed, such as adding the word »chapter« in front of the chapter number.

Because of this, a suitable mark for chapters might look like this:[5]

```
\renewcommand\chaptermark[1]{%
  \markboth{\memUChead{%
    \ifnum \c@secnumdepth >\m@ne
      \if@mainmatter
        \@chapapp\ \thechapter. \ %
      \fi
    \fi
     #1}}{}}%
```

How on earth would any novice (or even experienced) user be able to figure this out?

4. Using the \newlistof also create a new simple mark.
5. Slightly altered from the headings definition in *memoir*. \m@ne is short for −1 by the way.

By observing that more or less 90% of all users want dynamic contents within a quite short list of possibilities, we have added some new macros to *memoir* 2008:

```
\createplainmark{⟨type⟩}{⟨marks⟩}{⟨text⟩}
\createmark{⟨sec⟩}{⟨marks⟩}{⟨show⟩}{⟨prefix⟩}{⟨postfix⟩}
\addtopsmarks{⟨pagestyle⟩}{⟨prepend⟩}{⟨append⟩}
\nouppercasemarks \uppercasemarks
```

`\createplainmark` is used to create the plain marks, for example to create a suitable mark for the TOC, one can use

```
\createplainmark{toc}{both}{\contentsname}
```

which would automatically do the same as the more complicated

```
\renewcommand\tocmark{%
  \markboth{\memUChead{\contentsname}}{\memUChead{\contentsname}}}
```

`\createplainmark` is a lot easier to understand. The ⟨*type*⟩ refers to one of `toc`, `lot`,`lof`, `bib`, `index` and `glossary` with obvious meanings.[6]

⟨*marks*⟩ has to be of of »`both`« (meaning ⟨*text*⟩ is added to both the `\leftmark` and the `\rightmark`), »`left`« for just adding to the `\leftmark` (`\markboth{`⟨*text*⟩`}{}`) and »`right`« for just added to `\rightmark` (via `\markright{`⟨*text*⟩`}`). In general you would want to always use `both` for plain marks.[7]

The macro `\memUChead`, used internally, is a *memoir* macro that decides whether to convert the ⟨*text*⟩ in to upper case. The default definition of `\memUChead` is `\MakeUppercase`. One can redefine it as one likes. Though usually one would just issue `\nouppercaceheads` to disable `\memUChead` (i.e. redefine it to doing nothing), alternatively `\uppercaseheads` will make the ⟨*text*⟩ appear in uppercase. To remove the upper case part in a page style it is natural to add `\nouppercaceheads` to the start of ⟨*code*⟩ in `\makepsmarks`.

REMARK. *One may wonder why we do not just add the uppercase stuff to the styling for the header? This has something to do with expansion; it has turned out to be better to add it at the marks stage.*

* * *

6. Any use of `\newlistof` will add an extra plain mark.
7. Otherwise a long bibliography might be missing a header on even pages.

Setting up the regular plain marks is now as simple as:

```
\createplainmark{toc}     {both}{\contentsname}
\createplainmark{lof}     {both}{\listfigurename}
\createplainmark{lot}     {both}{\listtablename}
\createplainmark{bib}     {both}{\bibname}
\createplainmark{index}   {both}{\indexname}
\createplainmark{glossary}{both}{\glossaryname}
```

The sectional marks are a little more complicated. Let us first look at an example. The following code provides the *memoir* standard mark for chapters.[8]

```
\createmark{chapter}{left}{shownumber}{\@chapapp\ }{. \ }
```

Where `\@chapapp` is a *magic* macro defined as

$$\texttt{\textbackslash @chapapp} = \begin{cases} \texttt{\textbackslash appendixname} & \text{if we are within the appendix area} \\ \texttt{\textbackslash chaptername} & \text{else} \end{cases}$$

So by using `\chapapp` we do not have to change the page style when we enter the appendix area.

For `\createmark` the ⟨*sec*⟩ should be one of `chapter`, `section`, `subsection` or `subsubsection`.[9] Whereas ⟨*marks*⟩ is the same as with `\createplainmark`.

⟨*show*⟩ *has* to be either `shownumber` *or* `nonumber`; the effect will be discussed shortly.

The ⟨*prefix*⟩ and ⟨*postfix*⟩ are pre- and postfix texts that go before and after the sectional number. In the case of chapter marks, ⟨*prefix*⟩ is used to add the `\@chapapp` and some space before the chapter number; and ⟨*postfix*⟩ is used to add a dot and some space after the number.

The content to be provided to the page style by the sectional mark when set up via `\createmark` is determined according to this formula:[10]

8. Though I recommend using `both` instead of `left`.
9. Similar sectional macros like part, can also be used.
10. Here *sectional level* corresponds to the `secnumdepth` setting.

if ⟨*show*⟩ equals `shownumber`
    if current sectional level provides sectional numbers
        if within main matter
            ⟨*prefix*⟩⟨*sectional number*⟩⟨*postfix*⟩
        end if
    end if
end if
⟨*the provided title*⟩

By using these new macros, the style marks for the headings style can now be written

```
\makepsmarks{headings}{%
  \createmark      {chapter} {left} {shownumber}{\@chapapp\ }{. \ }
  \createmark      {section} {right}{shownumber}{}           {. \ }
  \createplainmark {toc}     {both} {\contentsname}
  \createplainmark {lof}     {both} {\listfigurename}
  \createplainmark {lot}     {both} {\listtablename}
  \createplainmark {bib}     {both} {\bibname}
  \createplainmark {index}   {both} {\indexname}
  \createplainmark {glossary}{both} {\glossaryname}
}
```

Wasn't that simple? It is easy to write and easy to understand—a clear improvement over earlier methods.

Remark. *What is up with the ⟨show⟩ option? What is this good for? It has been added to enable users who have, say, subsection numbers activated, but who do not want the subsection number to appear in the header. They should just use* `nonumber`.

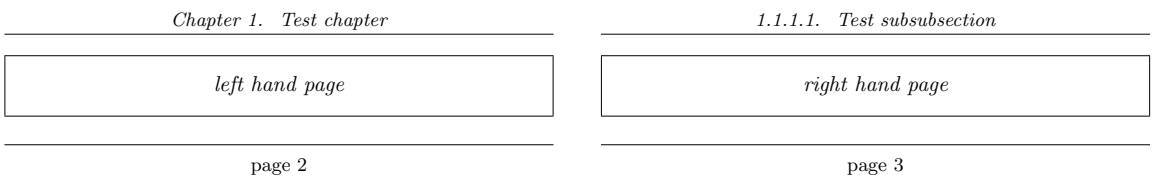
The last new macro, `\addtopsmarks{`⟨*pagestyle*⟩`}{`⟨*prepend*⟩`}{`⟨*append*⟩`}` can be used to extend a given `\makepsmarks{`⟨*style*⟩`}`. The prime example is to extend, say, headings to also include marks for sub- and subsubsection.

```
\addtopsmarks{headings}{}{
  \createmark{subsection}   {right}{shownumber}{}{. \ }
  \createmark{subsubsection}{right}{shownumber}{}{. \ }
}
```

2.5 An example

As a final example let us create a style like the ruled style from scratch. We would like to have the text in the center in both the header and footer.

```
\makepagestyle{myruled}
\makeheadrule {myruled}{\textwidth}{\normalrulethickness}
\makefootrule {myruled}{\textwidth}{\normalrulethickness}{\footruleskip}
\makeevenhead {myruled}{}{\small\itshape\leftmark} {}
\makeoddhead  {myruled}{}{\small\itshape\rightmark}{}
\makeevenfoot {myruled}{}{\small page \thepage}    {}
\makeoddfoot  {myruled}{}{\small page \thepage}    {}
\makeatletter % because of \@chapapp
\makepsmarks  {myruled}{
  \nouppercaseheads
  \createmark      {chapter}      {both} {shownumber}{\@chapapp\ }{. \ }
  \createmark      {section}      {right}{shownumber}{}           {. \ }
  \createmark      {subsection}   {right}{shownumber}{}           {. \ }
  \createmark      {subsubsection}{right}{shownumber}{}           {. \ }
  \createplainmark {toc}          {both} {\contentsname}
  \createplainmark {lof}          {both} {\listfigurename}
  \createplainmark {lot}          {both} {\listtablename}
  \createplainmark {bib}          {both} {\bibname}
  \createplainmark {index}        {both} {\indexname}
  \createplainmark {glossary}     {both} {\glossaryname}
}
\makeatother
\setsecnumdepth{subsubsection}
\pagestyle{myruled}
```



2.6 Conclusions

In section 1.4 on page 4 we listed some challenges; here are our conclusions to those challenges:

ad (1): The automatic upper casing feature has been factored out. Via \createplainmark and \createmark, the sectional marks, e.g. \chaptermark,

will internally use \memUChead as a wrapper macro around the text to be handed to the headers. \memUChead can be defined to use \MakeUppercase (the default) using \uppercaseheads or redefined to doing nothing via \nouppercaseheads. You could even redefine \memUChead to something completely different.

ad (2): In *memoir*, these settings have been separated out; and they will be set using plain mark macros such as \tocmark, \bibmark and \indexmark, all of which can now easily be set using \createplainmark.

ad (3): In *memoir*, \cleardoublepage (and similar macros) have been designed to issue the cleared page style on *blank* pages. This style *alias* initially points to empty but can be redefined to whatever the user wants it to be.

ad (4): The look of the headers and footers can easily be redefined using the *memoir* interface presented in section 2.3 on page 8.

ad (5): For the vast majority of all users the \createmark macro will provide all the flexibility users will ever need.

3 Scenarios

In this section we will list some (possible) real life scenarios and their solutions.

SCENARIO 1. *The headings style is* OK*, but we also want sub- and subsubsection titles in the headers, and please drop the all upper case thing.*

No problem.

```
\addtopsmarks{headings}{%
  \nouppercaseheads % added at the beginning
}{%
  \createmark{subsection}   {right}{shownumber}{}{. \space}
  \createmark{subsubsection}{right}{shownumber}{}{. \space}
}
% use the new settings
\pagestyle{headings}
\setsecnumdepth{subsubsection} % activating subsubsec numbering in doc
```

2 *Chapter 1. Test chapter* *1.1.1.1. Test subsubsection* 3

left hand page | *right hand page*

Scenario 2. *Requirements state that if a possible 'blank' page occurs before a new chapter, it should be in the normal page style.*

Assuming that headings is the style used, write:

```
\aliaspagestyle{cleared}{headings}
```

Scenario 3. *Create a style where the header is flushed towards the outer edge of the text block, containing the header information. Do the same for the footer which should contain the page number. Additionally, all other pages styles used should have their footer set in the same manner.*

The page style itself is straightforward; the 'hard' part is the requirement on the other used styles. But wait; in the standard setup all auxillary page styles that use page numbers are all aliases to plain or are the plain style itself, so it suffices to alter plain. To save a bit of space in this article we will base the new page style on headings and just use the marks that come with headings; insert your own replacements.

```
\copypagestyle{outer}{headings}
\makeoddhead{outer}{}{}{\slshape\rightmark}
\makeevenhead{outer}{\slshape\leftmark}{}{}
\makeoddfoot{outer}{}{}{\thepage}
\makeevenfoot{outer}{\thepage}{}{}
% fix plain
\copypagestyle{plain}{outer} % overwrite plain with outer
\makeoddhead{plain}{}{}{}    % remove right header
\makeevenhead{plain}{}{}{}   % remove left header
\pagestyle{outer}
```

CHAPTER 1. TEST CHAPTER *1.1. TEST SECTION*

left hand page	*right hand page*

2 3

Scenario 4. *On pages with only floats we would like to have no headers or footers.*

Here *memoir* has a trick up its sleeve: `\mergepagefloatstyle`; see [1], page 174, again assuming headings as the regular style used.

```
\mergepagefloatstyle{mergedstyle}{headings}{empty}
\pagestyle{mergedstyle}
```

Scenario 5. *The look of headings is* ok, *but please remove the section numbers from the header.*

```
\addtopsmarks{headings}{}{
  \createmark{section}      {right}{nonumber}{}{}
  \createmark{subsection}   {right}{nonumber}{}{}
  \createmark{subsubsection}{right}{nonumber}{}{}
}
\pagestyle{headings}
```

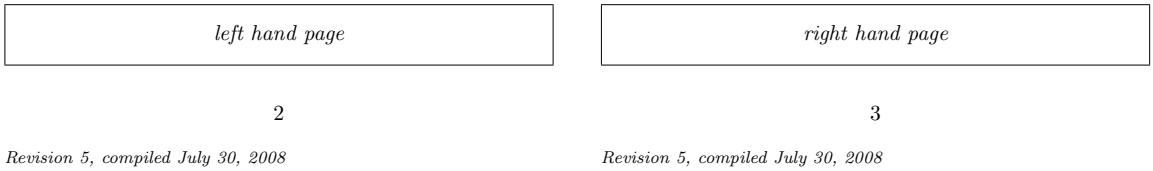
2 *CHAPTER 1. TEST CHAPTER* *TEST SUBSUBSECTION* 3

left hand page | *right hand page*

Scenario 6. *Add a revision number to all pages.*

There are several ways to do this; the easiest is to use the `eso-pic` package and use it to place the revision information onto the background. But we can also solve it using page styles. This requires creating a new page style (or modifying an existing one) and a rebuild of plain. The construction can be combined easily with other constructions to extract Subversion headers; see [2]. We will place the revision information below the footer to the left. We will modify headings but will show plain pages, so as to show the placement relative to the footer.

```
\newcommand\AddRevision{%
   \setlength\unitlength{1mm}
   \begin{picture}(0,0)
     \put(0,-7){\footnotesize\itshape Revision 5, compiled \today}
   \end{picture}}
\makeoddfoot{headings} {\AddRevision}{}{}
\makeevenfoot{headings}{\AddRevision}{}{}
\makeoddfoot{plain}    {\AddRevision}{\thepage}{}
\makeevenfoot{plain}   {\AddRevision}{\thepage}{}
%\pagestyle{headings}
\pagestyle{plain}
```

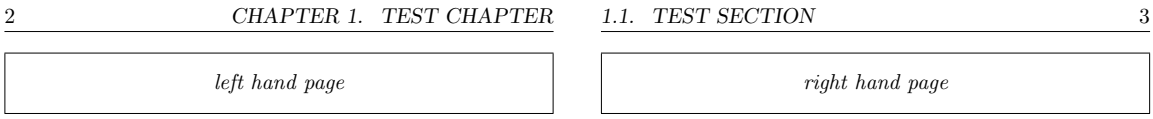


This could be extended in such a way that if, say, all chapters are in files of their own, then revision data for each separate chapter can be written at the bottom of the first page of each chapter. This is simply done by altering the chapter style, as above.

A colleague used the trick of using page styles to add other contents to the document, to implement a LaTeX version of our university letter style. It adds a colophon on the first page of a letter.

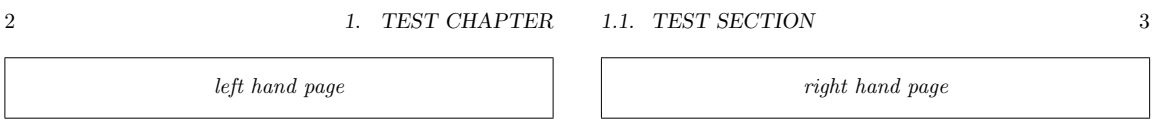
SCENARIO 7. *Add a line under the header.*

```
\makeheadrule{headings}{\textwidth}{\normalrulethickness}
\pagestyle{headings}
```



Scenario 8. *Remove the »Chapter« from the headings style.*

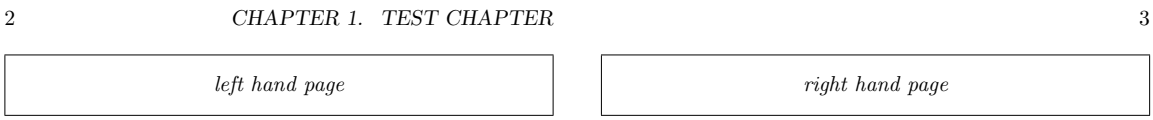
```
\addtopsmarks{headings}{}{
  \createmark{chapter}{both}{shownumber}{}{. \space}
}
\pagestyle{headings}
```



Scenario 9. *Remove the marks set by* `\section`.

This is a quite unusual request, so there is no nice interface for it. The easiest solution is

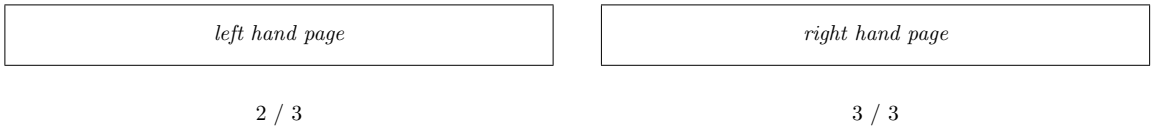
```
\addtopsmarks{headings}{}{%
  \renewcommand\sectionmark[1]{}
}
\pagestyle{headings}
```



Scenario 10. *I'd like to have a »page / total pages« footer. How do I implement that?*

In *memoir* we record two *total* numbers: the total number of sheets in the document, and the page number of the very last page. These are available via `\thelastsheet` and `\thelastpage`. Note that they are generally not the same, as the page number is usually reset to 1 at the beginning of the main matter. Let's change plain to solve the scenario:

```
\makeevenfoot{plain}{}{\thepage\ / \thelastpage}{}
\makeoddfoot {plain}{}{\thepage\ / \thelastpage}{}
\pagestyle{plain}
```



2 / 3 3 / 3

References

[1] Peter Wilson, *The Memoir Class for Configurable Typesetting – User Guide*, seventh edition, The Herries Press, 2008.

[2] PracTEX Journal, issue 2007-3. Available at http://www.tug.org/pracjourn/2007-3/index.html

[3] Lars Madsen, *Introduktion til LATEX*, preliminary version of the third edition, 2008. Available via http://www.imf.au.dk/system/latex/bog/.

[4] Frank Mittelbach and Michel Goossens, *The LATEX Companion*, Addison-Wesley, 2nd edition, 2004. ISBN 0-201-36299-6. With Johannes Braams, David Carlisle and Chris Rowley and contributions by Christine Detig og Joachim Schrod.

THE PracTEX Journal

Opinion: Enduring LaTeX documents

Lance Carnes

- Comment on this paper
- Send submission idea to editor

From PCTeXWiki

LaTeX documents that endure

The title of this opinion piece may seem a little strange. After all, if I keep my document source files safely stored away, and have a LaTeX system to format them, they should always work, right? Well, sometimes. More often than not, though, a set of LaTeX files more than a few years old will probably not format the same today as they did in the original edition.

Some examples

A common occurrence is a book author who is having difficulty making a new edition. A few years ago the author used LaTeX to format his or her book, and spent a lot of time getting the correct page breaks and figure placements. Now, when it's time to revise the book for a second edition, the author uses the same source files but finds that in many places the book does not break pages or float figures the same way as the original edition. There may even be LaTeX errors that definitely weren't there before.

Another situation that can be confusing is with authors who are collaborating on a book. Sometimes they find that they cannot format their source files to get a consistent rendering of their book. The two authors use the same source files, but Author A formats the book and gets different page breaks than Author B. In some cases one of the authors may get an "Undefined control sequence" or other error while trying to format the files sent by the other author.

In the first case the difference is that time has elapsed between attempts to format a document, and in the second case two LaTeX systems are being used in different locations and most likely on different computer platforms. In the first case the author's LaTeX system was updated, and something changed which caused different page breaks and figure floats. In the second case, the LaTeX systems used by the collaborating authors are probably of the same vintage but something is causing the authors to get different results.

LaTeX documents that endure

Anyone who has used LaTeX for a few years has a collection of document source files. Many of these older documents will format successfully with an up-to-date LaTeX system, but there will be some documents that will not look the same as the original edition, or they will cause LaTeX error messages. It sometimes takes a lot of work to get these older documents to format successfully again.

If the older documents are not critical, such as letters, homework assignments, or other material that is not published, the fact that these may not format correctly is not a problem. But for books and articles and other published material that may be revised and used again, it would save a lot of time if they could be formatted identically to the original edition.

LaTeX systems that work the same

When two or more authors are collaborating it would be best if the various LaTeX systems used could format the exact same document from the same source files. Another form of collaboration is a LaTeX consultant working with a client — it would save a

lot of time and expense if they each could reproduce the same exact document on their respective systems. But what often happens is that the LaTeX systems are different. For example, one person is using a Linux distribution and the other is using a PC distribution. The TeX programs that underlie the LaTeX systems on both platforms process files identically. However, formatting the same LaTeX source files on the the different platforms will often give different results.

Your ever-shifting LaTeX system

There are approximately 3,000 LaTeX system files. Some of these files are uniform throughout most LaTeX systems. The LaTeX "kernel" files and the common class files, such as article.sty, are identical on most LaTeX distributions. The files that are the cause of variability are the contributed class and style files. These files come into play when a document contains a `\documentclass` or `\usepackage` command. There are so many of these files and so many versions of each, that it is unlikely that any two LaTeX systems in the world contain the same set of LaTeX system files at any point in time.

Given all this possible variability in LaTeX system files, is it possible that documents can be formatted the same over time, and that LaTeX systems on different platforms can generate identical documents?

A current solution for enduring documents

One LaTeX system that can consistently format document editions is at Mathematical Science Publishers (http://mathscipub.com) at the University of California, Berkeley. Their journal issues and articles can be formatted identically to the original editions at any point in time. They accomplish this by using a *version control system* to guarantee that the original set of LaTeX system files used with a particular document is always used when formatting that document.

For further information on this system and its goals, see the Abstract, *Long-time preservation strategies for TeX-sourced content,* by Paulo Ney de Souza [1], and the video presentation [2]

Some other possibilities for enduring documents

One of the original goals of the TeX system is that given a valid TeX formatting program and a set of source files, the identical document can be formatted at any point in time and on any platform.

The problem is that the LaTeX system itself is a set of source files. When a LaTeX document is formatted, the TeX program first reads all the LaTeX system files and then inputs the author's source files. Since the TeX program is completely consistent, and the author controls his or her source files, the variable is the LaTeX system files.

One way to create an enduring document is to keep the current set of LaTeX system files with the author's source files. This guarantees that the set of input files is consistent over time. This is essentially how Mathematical Science Publishers is able to format documents consistently. Their version control system will pull the identical set of files from its archive each time a document is formatted.

Suppose there was a way that all LaTeX system files could be bundled together at a point in time, and kept along with the document source files. This would be one way to solve the problem of enduring documents. There might be some technical issues with organizing file system directories so that a LaTeX system could process the document. But if this problem could be solved it would provide a way to have enduring LaTeX documents.

One problem with this approach is that there would be a large cluster of files that could be used for one document only. This would work well for a single author working on a single computer system. However, if the author wanted to share this document with a collaborator, or send it to a publisher, there could be problems. The size of the file bundle would make this cumbersome, and it could be difficult for the person receiving the file bundle to get it all to work.

One way to reduce the size of the file cluster is to distribute the set of LaTeX system files to an online archive. Many current LaTeX systems fetch needed files from online archives. If the set of needed LaTeX system file versions could be kept in an index, a LaTeX system could use the index to fetch the needed files from an online archive. This would reduce the size of the file cluster since authors could exchange just their source document files along with an index of LaTeX system files.

In any case, it seems that the problem is one of ensuring that a unique set of files can be maintained so that a LaTeX document will

format identically over time and across computer systems. Once this problem is solved, it should be much easier and less time-consuming to maintain a set of documents that will format consistently.

What are your thoughts?

A few colleagues who read this piece agreed that consistently reproducible LaTeX documents is something that should be available to every author. There are various ways this could be done. Give it some thought and then let us know your ideas.

Some suggested that the LaTeX and TeX community, even with its slightly inconsistent set of system files, is probably better off than other documentation users. For example, how are authors with 15-year-old MS Word, InDesign, and other files, able to format the same document again?

Page generated August 24, 2008 ; TUG home page; search; contact webmaster.

THE PracTEX Journal

Travels in TeX Land: A bigger experiment with ConTeXt

David Walden

Abstract

In this column in each issue I muse on my wanderings around the TeX world. In my column in the 2007-2 issue I tried a small experiment with using ConTeXt. In this issue I describe an additional, quite extensive, effort to use ConTeXt -- to create a picture book for a "slide show" I was involved in creating a number of years ago.

David Walden is retired after a career as an engineer, engineering manager, and general manager involved with research and development of computer and other high tech systems. More information is available at www.walden-family-com. He may be contacted at dave@walden-family.com. A complete list of his "travels in TeX Land" pieces is at http://www.walden-family.com/dave/index.htm#travels-in-texland.

- PDF version of paper
- scorton-marsh-script.txt
- book-initial.tex
- book-initial.pdf
- slides2tex.pl
- scorton-marsh-script.tex
- book-in-context.tex
- book-in-context.pdf
- book.tex
- book-example.pdf
- Comment on this paper
- Send submission idea to editor

The PracTEX Journal, 2008, No. 2
Article revision 2008/6/20

Travels in TEX Land: A bigger experiment with ConTEXt

David Walden

Abstract In this column in each issue I muse on my wanderings around the TEX world. In my column in the 2007-2 issue (http://www.tug.org/pracjourn/2007-2/walden/) I tried a small experiment with using ConTEXt. In this issue I describe an additional, quite extensive, effort to use ConTEXt — to create a picture book for a "slide show" I was involved in creating a number of years ago.

Background regarding the book

I live on the edge of a salt marsh on Cape Cod in Massachusetts. In 1991 my son, a photographer, and I undertook a two week summer vacation project to study the marsh. He took the 35mm photos and I was the photographers assistant. We both did research about the marsh in the library of our local nature center. We wrote a script and recorded it on cassette tape. And on the second Saturday of our vacation, after sunset when it was dark enough for the slides to show great colors, we presented our slide show of the marsh to my wife, manually changing the slides to follow the tape recorded narrative.

In the following years, I presented the slide show on several occasions to small groups of guests in our home. Eventually I got the idea that I should create a CD version of the slide show that I could give to people to watch on their own computers.

In about 2000 I paid Kodak to scan the 212 slides in our slide show and to give me a CD with high resolution digital versions of the slides. In 2005 I finally got around to creating a CD of the slide show, using basic HTML.

First I processed every digital slide with Photoshop to create cropped, color-adjusted, low resolution `.jpg` files of each slide suitable for computer screen display without taking up as much space as the scans from Kodak did. Next, I

Copyright © 2008 David Walden

edited the Word file of the script of the slide show to a a driver file which could be processed with a Perl program; see `scorton-marsh-script.txt` on the HTML page for this column. Notice the code (p1 through p6) in the third argument of each entry in the drive file which specifies how to format different styles of HTML pages depending on the orientation of the image, pages with section titles, and so forth.

The Perl program generated an `.html` file with navigation links; you can see an example of a few "pages" of such output at `marsh-slideshow/001.htm`. I copied a directory containing `.jpg` and .html files onto a CD, printed the CD art on a circular adhesive label, pasted the label on the CD, and gave people the CD with instructions of how to click on the first `.html` page in sequence on the CD.

Recently I have been experimenting with self-publishing (as described in several of my columns), and I decided that it would be nice to create and publish a picture book from the slide show. This is the subject of the rest of this column.

Initial thoughts on the book

My first mental image of the picture book had a marsh photo on each page at a specific spot relative to the top and side margins with the text for the page in a separate place on the page. For this it seemed to me that I needed to avoid using most of TEX's float capability and ideally would have several templates that would determine the layout of each of several types of pages. I guessed this would be easier to do in ConTEXt than LATEX, although this was not a well-informed guess. In any case, I decided to produce the book using ConTEXt—it couldn't hurt to get some significant experience with ConTeXt.

I asked Aditya Mahajan (who is writing a series of articles on ConTEXt for *TUGboat*) whether ConTEXt had the template capability that I envisioned, and he directed me to ConTEXt's layers capability: http://wiki.contextgarden.net/Layers. With that assurance that I could do what I wanted with ConTEX, I began to create a ConTEXt file that had a book format. The first, sketchy effort, is documented in the files `book-initial.tex` and `book-initial.pdf`.

Creating the basic ConTEXt program

At this point, it seemed like it would be useful to generate a ConTEXt program for the entire book which generated appropriate macro calls for each type of book page based on my existing `scorton-marsh-script.txt` file. In other words, it was time to modify my Perl program, which seemed a relatively easy prospect since all it had to do was generate a macro call for each type of output page and I could define the macro calls later in ConTEXt (and, thus, modifying the program seemed mostly to involve skipping over code to output various lines of HTML).

In fact, it took me about an hour to modify the program; see `slides2tex.pl` if you are interested. With an an almost TEX program in hand, I manually modified the file to remove angle brackets, to at least initially try to capture section titles embedded in type two slides, etc.; this took another hour to get a TEX file that PDFTEX could compile: `scorton-marsh-script.tex`. I figured that if PDFTEX could compile the file, then I was not too far from something ConTEXt could compile.

If you look at that TEX file, you will see that each type of slide went to the same macro definition. Now it was time to define separate macros for each slide type — and to figure out some actual lines of ConTEX for these macros. I merged the contents of textttscorton-marsh-script.tex into my `book-initial.tex` ConTEXt file, called the merged version `book-in-context.tex`, and made a few necessary trivial modifications to the script part.

Within another hour or two I had compiled the draft book in ConTEX; see `book-in-context.tex` and `book-in-context.pdf` for an example of this version before detailed formatting was begun.

The power of the command-based approach to typesetting of TEX is never more apparently than when TEX is used for a project like this which involves converting content from a markup language in an existing file and the pages of the document can be described with only a few templates.

By the way, my approach to learning what I needed to know about ConTEXt to get this far was to go to `contextgarden.net` and search for words involved with whatever I needed to know or to search Hans Hagen's *ConTEXt Manual* at `http://www.pragma-ade.com/show-man-9.htm` or other manuals at `http://www.pragma-ade.com/show-man-1.htm`.

Refining the ConTEXt program

Next I needed to deal with the detailed formatting in ConTEXt and to return to my high resolution scans from Kodak and reprocessing them to get print quality images. Such reprocessing of the images to get appropriate cropping and good looking colors took me a several hours a day over two or three weeks the last time I did it, so I started with the formatting issues.

I wanted to have a little space between paragraphs of text on the photo pages. Using the `\setupwhitespace` put a space between paragraphs but I was unable to control the height of the space with the argument of the command; the space was always the same size for the parameter values I selected. Then I discovered that I had to turn off typesetting on a grid, which was in a command left over from the file for my earlier column which was my starting point for this project.

I noticed that ConTEXt was numbering pages starting with 1 for the cover page of the book and, thus, the first page after the front matter was numbered 4. The front matter pages didn't have page numbers which was OK—I had no need to give them page numbers using small Roman numberals. I searched `contextgarden.net` and found the command `\setnumber[page]{1}` which I added after the front matter in the ConTEXt file.

I wanted to use PDFTEX's capability to do micro-typesetting, both protrusion beyond the end of the line and font expansion and contraction. However, I couldn't find anything about micro-typesetting by searching the `contextgarden.net` or the *ConTEXt Manual*. Eventually Aditya Mahajan directed me to `http://wiki.contextgarden.net/Protrusion` which gives various synonyms for micro-typesetting but doesn't use that term exactly. Eventually I found a combination of commands that effected micro-typesetting:

```
%\usetypescript[serif][hz][quality]
%\setupalign[hanging,hz]
\usetypescript[serif,sans,mono][hanging][pure]
\setupalign[hanging]
\usetypescript[modern-base][texnansi]
\setupbodyfont[reset]
\setupbodyfont[modern]
```

These commands did slightly different micro-typesetting on a few page test case with the first two lines above commented in and the next two lines commented

out. However, when I tried either sequence of commands with the entire book, pdfTEX came to a fatal error saying it had run out of space and something more about a hash table. With a little testing I discovered that commenting out the last three lines above let the whole book compile, but then I didn't appear to get micro-typesetting. When I asked the ConTEXt mailing list (`ntg-context@ntg.nl`) about this, Hans Hagen suggested that I "increase the hash size — this is done in `texmf.cnf`; after doing that you need to remake the formats." However, I could not find `texmf.cnf` on my computer (I am using whatever version of ConTEXt came with the version of ProTEXt from a couple of years ago), and I didn't know what "remake the formats" meant. I have little interest ever in struggling with configuration issues, and definitely didn't want to hear that I needed to get a newer version of ConTEXt, so I gave up on using micro-typesetting; instead I changed the text wording in the places in the sparse text of this picture book to eliminate the need for micro-typesetting.

By the way, I do all of my ConTEX compiling using `texexec`. I also always wish that ConTEXt had a different name so that Googling on it and some other keywords didn't find all of the instances of the keywords in whatever "context" they appear.

My next question was how to get an even number of pages in the front matter such that page 1 of the main matter started on an odd numbered page. Another search of `contextgarden.net` (for "double-sided") took me to a discussion of FO (whatever that is) page layout including the command

```
\setuppagenumbering[alternative=doublesided, state=stop]
```

I didn't know what the `state=stop` part of the command means, so I deleted it and the remaining command seemed to do what I wanted it to do. It also started each new chapter on an odd numbered page, and considered the possibility of turning off double-sided page numbering for the main body of the book — a final decision which could be made later.

Of course, every change can bring up new issues, for example, with double-side page number, how could I leave an even numbered page at the end of a chapter completely blank — no page number. After searching `contextgarden.net` and the manuals at `www.pragma-ade.com` for an hour or so without figuring out how to do this, I sent a message to the `ntg-context@ntg.nl` mailing list. I immediately received a message back from Jörg Hagmann indicating I should learn

about learn about \definepagebreak. I searched contextgarden.net and the archives of ntg-context@ntg.nl (http://www.ntg.nl/pipermail/ntg-context/) for \definepagebreak, found a long discussion about what worked and didn't work for various people, but I couldn't make any of those ideas work and eventually gave up. Instead I simply changed my macro for the start of chapters so it does a \setuppagenumbering[state=stop] before the \chapter call to turn off page numbering for any blank page generated at the end of the previous chapter and does a \setuppagenumbering[state=start] after some text for the chapter has been generated which turns page numbering back on for the first page of the chapter (which is how I want it to be).

However, despite how I left things at the end of the last paragraph, Aditya Mahajan sent another response to the ConTEXt mailing list asking me if the following commands (which I have paraphrased to match my actual situation) had the right result.

```
\definepagebreak
  [mychapterpagebreak]
  [yes,header,footer,right]

\setuphead
  [chapter]
  [page=mychapterpagebreak]

\setuppagenumbering[alternative=doublesided,location=footer]
\setupfootertexts[{\ssx {\pagenumber}}]

\starttext
  \chapter {testA} \dorecurse{10}{\input tufte }
  \chapter {testB} \dorecurse{10}{\input tufte }
  \chapter {testC} \dorecurse{10}{\input tufte }
\stoptext
```

I tried the above test case and it worked, so I adapted it to my actual book file and removed the start- and stop-numbering commands I described at the end of the last paragraph.

I cannot avoid any longer the giant job of reprocessing my original scans for Kodak. Thus, I'll stop working with ConTEXt for now. The current source file for

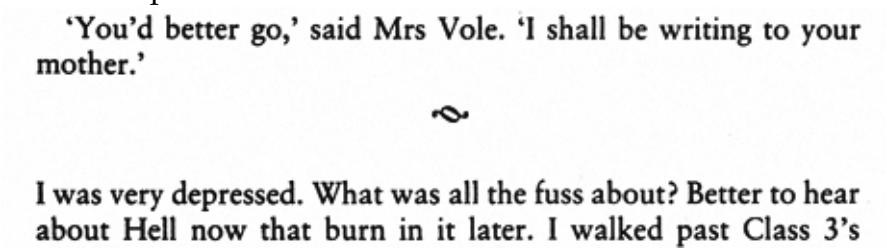
my book is `book.tex`, and an example of a few pages from the typeset book is `book-example.pdf`. I'll try to remember to update this column with information on how to buy the printed book when it is available for sale.

Follow-up on my prior column

In my column in TPJ issue 2008-1, I once again discussed what I call "thought breaks," which I previously had discussed in issue 2005-4.

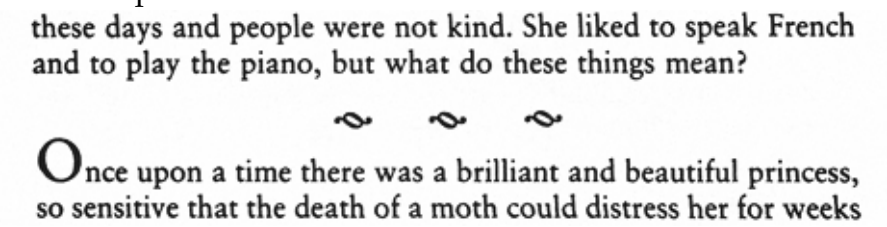
I am becoming more and more sensitive to these ornaments used for typesetting such thought breaks. A few weeks ago I came upon another ornament that I particularly like — in Jeanette Winterson's book *Oranges Are Not the Only Fruit*, Grove Press, New York, 1987. This book uses a variation of the same symbol for smaller and bigger thought breaks, as follows.

Smaller break example:


'You'd better go,' said Mrs Vole. 'I shall be writing to your mother.'

I was very depressed. What was all the fuss about? Better to hear about Hell now that burn in it later. I walked past Class 3's

Bigger break example:


these days and people were not kind. She liked to speak French and to play the piano, but what do these things mean?

Once upon a time there was a brilliant and beautiful princess, so sensitive that the death of a moth could distress her for weeks

Acknowledgments

Aditya Mahajan answered key questions about using ConTEXt for this project and provided other guidance. Responses from participants in the ConTEXt mailing list were also helpful. Yuri Robbers made several suggestions for improving the final manuscript.

Biographical note

David Walden is retired after a career as an engineer, engineering manager, and general manager involved with research and development of computer and other high tech systems. He holds an undergraduate math degree and completed a graduate school sequence of courses in computer science. More history is at www.walden-family.com/dave.

THE PracTEX Journal

Ask Nelly:
How do I center only the last line of a paragraph?
How do I get the first and last entry of each index page in its header?

The Editors

Abstract

Ask Nelly is a question and answer column. Nelly is the quiet person who sits at the back corner desk, who knows a lot, and when asked any question is always ready with a patient answer. If Nelly doesn't know the answer, Nelly will know an expert who has the answer. Feel free to Ask Nelly about any aspect of LaTeX, TeX, Context, etc.

- Comment on this paper
- Send submission idea to editor

TPJ

Q: Dear Nelly: Sometimes I need to write centered paragraphs of text. I want the paragraph to be fully justified, but the last line to be centered. How do I achieve that? I work in Plain TeX.

A: Normally centering a paragraph is achieved by setting equal `stretch` and `shrink` amounts for `\leftskip` and `\rightskip` and setting `\parindent` and `\parfillskip` to zero.

A slight variation on this theme fully justifies all lines except the last one, which will be centered.

```
\leftskip = 0cm plus 0.5fil
\rightskip = 0cm plus -0.5fil
\parfillskip = 0cm plus 1fil
```

The stretch components for all lines of a paragraph except the last one are 0, so zero space is inserted by `\leftskip` and `\rightskip`. On the last line of the paragraph the `\parfillskip` adds `1fil` of stretch, bringing the total for both ends of the line to `0.5fil` and thus centering the line.

The above question was answered by **Yuri Robbers**, a member of the editorial board of this journal. He can be reached at `yuri.robbers@gmail.com`

TPJ

Q: Dear Nelly: I'd like to have the page headers for my index list the first and last entry on the page, for easier reference. Dictionaries and encyclopaedias tend to do this too. How can I do this in LaTeX?

A: To achieve this, we will use the `fancyhdr` package by Piet van Oostrum and `makeindex` by Pehong Chen, both available on most TeX systems and all CTAN mirrors.

First we define a page style specifically for the index:

```
\pagestyle{index}
\fancyhead{}
\fancyhead[RO,LE]{\rightmark -- \leftmark}
```

Then we define a new command

```
\newcommand{\indexmark}[1]{#1\markboth{#1}{#1}}
```

Then we add the following lines to the configuration file for `makeindex`

```
item_0 "\n\\item \\indexmark{"
delim_0 "},"
```

And if you ever upgrade to encyclopaedias or dictionaries you may run into a definition that spans more than one page. In that case you may want to replace the last line op the page style definition with

```
\fancyhead[RO,LE]{%
  \ifthenelse{\equal{\leftmark}{\rightmark}}%
    {\rightmark}%
    {\rightmark -- \leftmark}%
}
```

And then of course do not forget to issue a

```
\pagestyle{index}
```

at the beginning of your index.

The above question was answered by **Yuri Robbers**, a member of the editorial board of this journal. He can be reached at yuri.robbers@gmail.com

TPJ

THE PracTEX Journal

TEX USERS GROUP

Distractions: Fun packages — sudoku solvers

The Editors

- Comment on this paper
- Send submission idea to editor

Sudoku Solvers

The `sudokubundle` and `lasudoku` styles are ways to have some fun with LaTeX, and at the same time learn more about how packages are used and developed. Both packages were created based on a challenge in an early issue of The PracTeX Journal. `sudokubundle` was created by Peter Wilson of Herries Press, and and `lasudoku` was written by Zachary Catlin of Purdue University. We will show how to use these packages to print a sudoku puzzle, and solve it. These packages use LaTeX only, with no help from outside programs or scripts.

In the 2005-4 issue of The PracTeX Journal the editors issued a challenge to create a TeX-based sudoku solver. You can see the challenge and some background on sudokus at http://tug.org/pracjourn/2005-4/distract/#sudoku. There were three contest levels, and the expert level asked for a TeX-based document that would read a sudoku puzzle file, format it, and then solve it. There were two contest entrants who sent working LaTeX-based solvers, Peter Wilson and Zachary Catlin. You can review their submissions at http://tug.org/pracjourn/2006-1/distract/sudoku_winners/answers.html.

The solvers introduce two commands, `\sudoku` and `\sudokusolve`. Each command accepts an argument, the name of a sudoku position file, and then prints and solves it. Zachary Catlin's solver is a LaTeX package called `lasudoku`. Peter Wilson's original submission was a LaTeX document that contained the sudoku printing and solving algorithms, but was not organized as a package. Later he developed this into a full package, and added a third utility to generate new puzzles. If you are interested in how a set of tools is eventually organized into a package, look at Peter's first effort and compare it with the eventual `sudokubundle` package.

In this article we will be using Peter Wilson's `sudokubundle` package, which was contributed to the TeX Archive (CTAN) and is part of the TeX Live distribution. Similar results can be had by using Zachary Catlin's `lasudoku` package. You may want to try both solvers and compare results. Keep in mind that they are able to solve a large number of easy and medium difficulty puzzles, but may not be able to solve all puzzles.

Download the sample files Unzip this file into a directory. Format `sudokubundleTest.tex` or `lasudokuTest.tex` with LaTeX to see how the packages work.

sudokubundle documentation

The starting position

To test these packages, find a sudoku puzzle to print or solve. Look for puzzles in your local newspaper or go to an online sudoku site, for example http://www.sudokusolver.co.uk. Here is a sample puzzle we will use in this article:

6	4			1	3	9		
1				2	6	4		
	2	9		4	5	7		
		2				8	3	
8	6			3	7		1	9
7			2		9			
		1	3			6	9	
9	3	6	4		8		2	
		5						

In order to print or solve a puzzle, it must be in a form that `sudokubundle` can use. One way to do this is to create a simple text file with the starting position.

```
64..139..
1...264..
.29.457..
..2...83.
86..37.19
7..2.9...
..13..69.
9364.8.2.
..5......
Easy sample sudoku
from http://www.sudokusolver.co.uk
```

There are nine characters per line and nine lines, corresponding to the 9x9 sudoku grid. The numbers 1-9 are filled-in squares, and a period represents a blank square. After the ninth line you can add anything you want.

Write this text to a file. For this article we have created the file `SudokuSolverSample.sud`, which you can find in the sample files at the top of this article.

(There is another way to enter starting positions into the `sudokubundle` package. See the package documentation for this method, and for complete package information.)

Printing a puzzle

This requires just one command: `\sudoku{SudokuSolverSample.sud}`. The argument for this command, `SudokuSolverSample.sud`, is the file we made in the previous section. Here is a sample LaTeX document you can use to print a sudoku:

```
% Test the sudokubundle package
\documentclass{article}

\usepackage{createsudoku}

\begin{document}

\begin{center}
\sudoku{SudokuSolverSample.sud} % Print a puzzle
\end{center}

\end{document}
```

Format this document with LaTeX and it will print a puzzle position like this:

6	4			1	3	9		
1				2	6	4		
	2	9		4	5	7		
		2				8	3	
8	6			3	7		1	9
7			2		9			
		1	3			6	9	
9	3	6	4		8		2	
		5						

Solving a sudoku

This requires just one command: `\sudokusolve{SudokuSolverSample.sud}`. The argument for this command, `SudokuSolverSample.sud`, is the file we made in the first section. In the example LaTeX document above, substitute the following line for `\sudoku{SudokuSolverSample.sud}`.

```
\sudokusolve{SudokuSolverSample.sud} % Solve a puzzle
```

Run this document through LaTeX and it will print the puzzle as above followed by the solution:

THE ANSWER

6	4	8	7	1	3	9	5	2
1	5	7	9	2	6	4	8	3
3	2	9	8	4	5	7	6	1
5	9	2	1	6	4	8	3	7
8	6	4	5	3	7	2	1	9
7	1	3	2	8	9	5	4	6
4	7	1	3	5	2	6	9	8
9	3	6	4	7	8	1	2	5
2	8	5	6	9	1	3	7	4

For those who want to dissect the package, a good place to start is by examining the `.log` file, where the package writes notes as it solves puzzles.

Creating a sudoku

The `sudokubundle` package can generate a new puzzle from a solved puzzle. The command to do this is `\generategrid [SudokuSolverSampleSolution.sud]`. Note that the file name is optional and is in square brackets. The file is a sudoku position that was created the same way as in the first section, except that this file must be a valid solution position. You can type the position into a text file and save it.

However, there is another way to create a new puzzle without typing a solution file. Let the package solve a puzzle, and then let it create a new puzzle from this solved puzzle. (It writes the solution to a default solution file, `sud.out`, and then uses this position file.) Use the following sample LaTeX document to create a new puzzle:

```
% Test the sudokubundle package
\documentclass{article}

\usepackage{createsudoku}

\begin{document}

\begin{center}
\sudokusolve{SudokuSolverSample.sud} % Solve a puzzle
\generategrid  % Use this solution to generate a new puzzle
\end{center}

\end{document}
```

Surprising that LaTeX can solve a sudoku

When the original suduko challenge was proposed, one of the editors wrote that he didn't think anyone would submit a solution. At first glance, it didn't even seem possible to write a TeX or LaTeX document that could do something that complex. But, in fact, TeX (LaTeX) is Turing complete which means it can solve any problem that other computing devices can solve. However, TeX isn't always the best tool to use to solve complex problems --- for specialized problems there are usually better tools. In any case, it's nice to know that the problem-solving power is there if you need it!

Retrieved from "http://wiki.pctex.com/index.php/Distract-Sudoku"

Page generated August 24, 2008 ; TUG home page; search; contact webmaster.